Normalization

- Normalization is the process of organizing data in a database.
- It includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy.

***Redundant data wastes disk space and creates maintenance problems.

- If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations.
- A customer address change is easier to implement if that data is stored only in the Customers table and nowhere else in the database.
- There are a few rules for database normalization.
- Each rule is called a "normal form."
- If the first rule is observed, the database is said to be in "first normal form."
- If the first three rules are observed, the database is considered to be in "third normal form."
- Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

As we have discussed above, normalization is used to reduce data redundancy. It provides a method to remove the following anomalies from the database and bring it to a more consistent state:

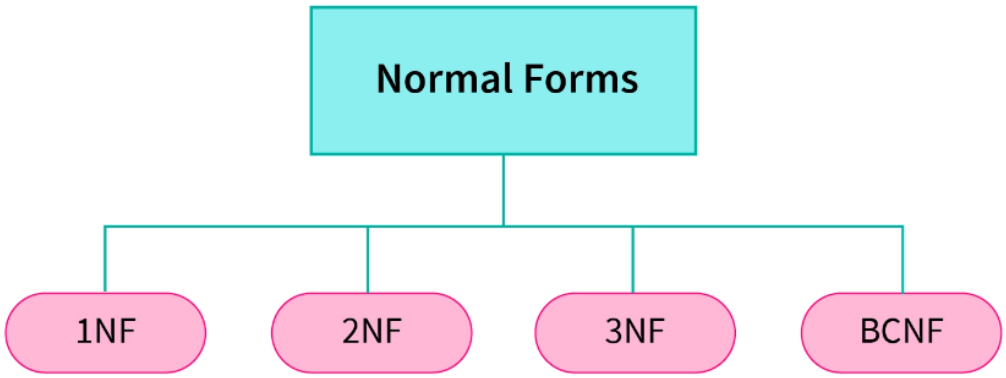
A database anomaly is a flaw in the database that occurs because of poor planning and redundancy.

1. **Insertion anomalies**: This occurs when we are not able to insert data into a database because some attributes may be missing at the time of insertion.
2. **Updation anomalies:** This occurs when the same data items are repeated with the same values and are not linked to each other.
3. **Deletion anomalies:** This occurs when deleting one part of the data deletes the other necessary information from the database.

vidyasagar.bandaluppi@gmail.com

Normal Forms

There are four types of normal forms that are usually used in relational databases as you can see in the following figure:



1. **1NF:** A relation is in 1NF if all its attributes have an atomic value.
2. **2NF:** A relation is in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the candidate key in DBMS.
3. **3NF:** A relation is in 3NF if it is in 2NF and there is no transitive dependency.
4. **BCNF:** A relation is in BCNF if it is in 3NF and for every Functional Dependency, LHS is the super key.

To understand the above-mentioned normal forms, we first need to have an understanding of the functional dependencies.

Functional dependency is a relationship that exists between two sets of attributes of a relational table where one set of attributes can determine the value of the other set of attributes. It is denoted by X -> Y, where X is called a determinant and Y is called dependent.

There are various levels of normalizations. Let's go through them one by one:

First Normal Form (1NF)

A relation is in 1NF if every attribute is a single-valued attribute or it does not contain any multi-valued or composite attribute, i.e., every attribute is an atomic attribute. If there is a composite or multi-valued attribute, it violates the 1NF. To solve this, we can create a new row for each of the values of the multi-valued attribute to convert the table into the 1NF.

Let's take an example of a relational table <EmployeeDetail> that contains the details of the company.

<EmployeeDetail>

Employee Code	Employee Name	Employee Phone Number
101	John	98765623,998234123
101	John	89023467
102	Ryan	76213908
103	Stephanie	98132452

Here, the Employee Phone Number is a multi-valued attribute. So, this relation is not in 1NF.

To convert this table into 1NF, we make new rows with each Employee Phone Number as a new row as shown below:

<EmployeeDetail>

Employee Code	Employee Name	Employee Phone Number
101	John	998234123
101	John	98765623
101	John	89023467
102	Ryan	76213908
103	Stephanie	98132452

Second Normal Form (2NF)

vidyasagar.bandaluppi@gmail.com

The normalization of 1NF relations to 2NF involves the elimination of partial dependencies. A partial dependency in DBMS exists when any non-prime attributes, i.e., an attribute not a part of the candidate key, is not fully functionally dependent on one of the candidate keys.

For a relational table to be in second normal form, it must satisfy the following rules:

1. The table must be in first normal form.
2. It must not contain any partial dependency, i.e., all non-prime attributes are fully functionally dependent on the primary key.

If a partial dependency exists, we can divide the table to remove the partially dependent attributes and move them to some other table where they fit in well.

Let us take an example of the following <EmployeeProjectDetail> table to understand what is partial dependency and how to normalize the table to the second normal form:

<EmployeeProjectDetail>

Employee Code	Project ID	Employee Name	Project Name
101	P03	John	Project103
101	P01	John	Project101
102	P04	Ryan	Project104
103	P02	Stephanie	Project102

In the above table, the prime attributes of the table are Employee Code and Project ID. We have partial dependencies in this table because Employee Name can be determined by Employee Code and Project Name can be determined by Project ID. Thus, the above relational table violates the rule of 2NF.

The prime attributes in DBMS are those which are part of one or more candidate keys.

To remove partial dependencies from this table and normalize it into second normal form, we can decompose the <EmployeeProjectDetail> table into the following three tables:

<EmployeeDetail>

Employee Code	Employee Name
101	John
101	John
102	Ryan
103	Stephanie

<EmployeeProject>

Employee Code	Project ID
101	P03
101	P01
102	P04
103	P02

<ProjectDetail>

Project ID	Project Name
P03	Project103
P01	Project101
P04	Project104
P02	Project102

vidyasagar.bandaluppi@gmail.com

Thus, we've converted the <EmployeeProjectDetail> table into 2NF by decomposing it into <EmployeeDetail>, <ProjectDetail> and <EmployeeProject> tables. As you can see, the above tables satisfy the following two rules of 2NF as they are in 1NF and every non-prime attribute is fully dependent on the primary key.

The relations in 2NF are clearly less redundant than relations in 1NF. However, the decomposed relations may still suffer from one or more anomalies due to the transitive dependency. We will remove the transitive dependencies in the Third Normal Form.

The normalization of 2NF relations to 3NF involves the elimination of transitive dependencies in DBMS.

A functional dependency X -> Z is said to be transitive if the following three functional dependencies hold:

- X -> Y
- Y does not -> X
- Y -> Z

For a relational table to be in third normal form, it must satisfy the following rules:

1. The table must be in the second normal form.
2. No non-prime attribute is transitively dependent on the primary key.
3. For each functional dependency X -> Z at least one of the following conditions hold:

- X is a super key of the table.
- Z is a prime attribute of the table.

If a transitive dependency exists, we can divide the table to remove the transitively dependent attributes and place them to a new table along with a copy of the determinant.

Let us take an example of the following <EmployeeDetail> table to understand what is transitive dependency and how to normalize the table to the third normal form:

<EmployeeDetail>

Employee Code	Employee Name	Employee Zipcode	Employee City
101	John	110033	Model Town
101	John	110044	Badarpur
102	Ryan	110028	Naraina
103	Stephanie	110064	Hari Nagar

The above table is not in 3NF because it has Employee Code -> Employee City transitive dependency because:

- Employee Code -> Employee Zipcode
- Employee Zipcode -> Employee City

Also, Employee Zipcode is not a super key and Employee City is not a prime attribute.

To remove transitive dependency from this table and normalize it into the third normal form, we can decompose the <EmployeeDetail> table into the following two tables:

<EmployeeDetail>

Employee Code	Employee Name	Employee Zipcode
101	John	110033
101	John	110044
102	Ryan	110028
103	Stephanie	110064

<EmployeeLocation>

Employee Zipcode	Employee City
110033	Model Town
110044	Badarpur
110028	Naraina
110064	Hari Nagar

Thus, we've converted the <EmployeeDetail> table into 3NF by decomposing it into <EmployeeDetail> and <EmployeeLocation> tables as they are in 2NF and they don't have any transitive dependency.

The 2NF and 3NF impose some extra conditions on dependencies on candidate keys and remove redundancy caused by that. However, there may still exist some dependencies that cause redundancy in the database. These redundancies are removed by a more strict normal form known as BCNF.

Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form(BCNF) is an advanced version of 3NF as it contains additional constraints compared to 3NF.

For a relational table to be in Boyce-Codd normal form, it must satisfy the following rules:

1. The table must be in the third normal form.
2. For every non-trivial functional dependency X -> Y, X is the superkey of the table. That means X cannot be a non-prime attribute if Y is a prime attribute.

A superkey is a set of one or more attributes that can uniquely identify a row in a database table.

Let us take an example of the following <EmployeeProjectLead> table to understand how to normalize the table to the BCNF:

<EmployeeProjectLead>

Employee Code	Project ID	Project Leader
101	P03	Grey
101	P01	Christian
102	P04	Hudson
103	P02	Petro

The above table satisfies all the normal forms till 3NF, but it violates the rules of BCNF because the candidate key of the above table is {Employee Code, Project ID}. For the non-trivial functional dependency, Project Leader -> Project ID, Project ID is a prime attribute but Project Leader is a non-prime attribute. This is not allowed in BCNF.

To convert the given table into BCNF, we decompose it into three tables:

<EmployeeProject>

Employee Code	Project ID
101	P03
101	P01
102	P04
103	P02

<ProjectLead>

Project Leader	Project ID
Grey	P03
Christian	P01
Hudson	P04
Petro	P02

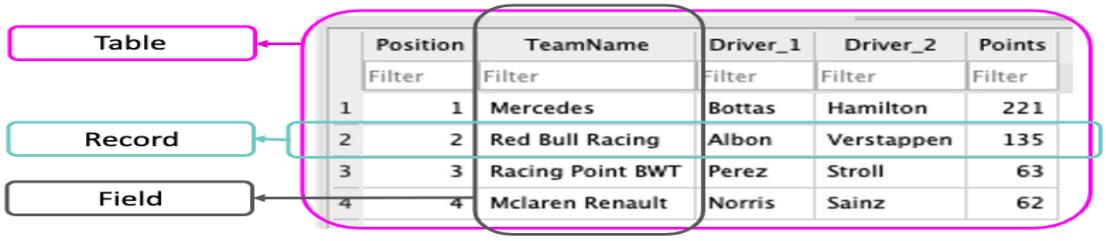
Thus, we've converted the <EmployeeProjectLead> table into BCNF by decomposing it into <EmployeeProject> and <ProjectLead> tables.

vidyasagar.bandaluppi@gmail.com

DBMS types :

Flat file

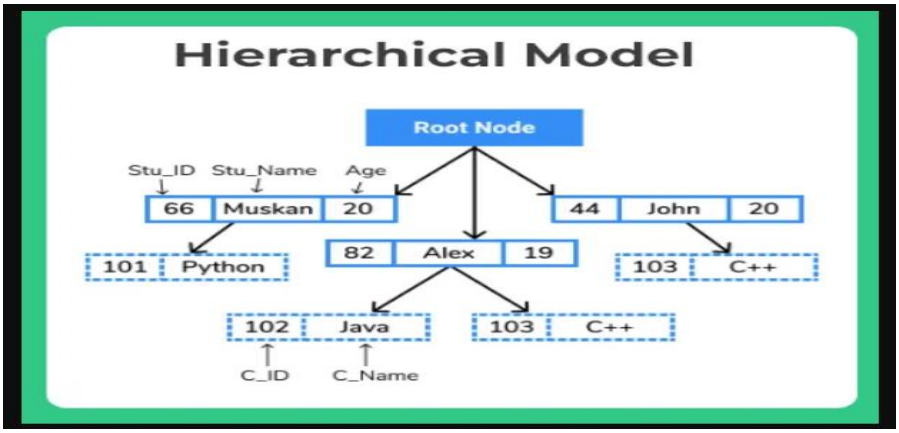
A flat file database is a database where all the records are stored in a single table in a non-database software. Ex. MS-Excel, CSV files.



Hierarchical model in DBMS

- Hierarchical data model is being used from the 1960s onwards where data is organized like a tree structure
- In 1966 IBM introduced an information management system(IMS product) ***which is based on this hierarchical data model but now it is rarely used.***

Hierarchical model:
The hierarchical model organizes the data into ***a tree structure which consist of a single root node where each record is having a parent record and many child records and expands like a tree***



XML data

Data is represented in the form of tags. XML data is compatible to access across technologies like Java, DotNet, Python, PHP

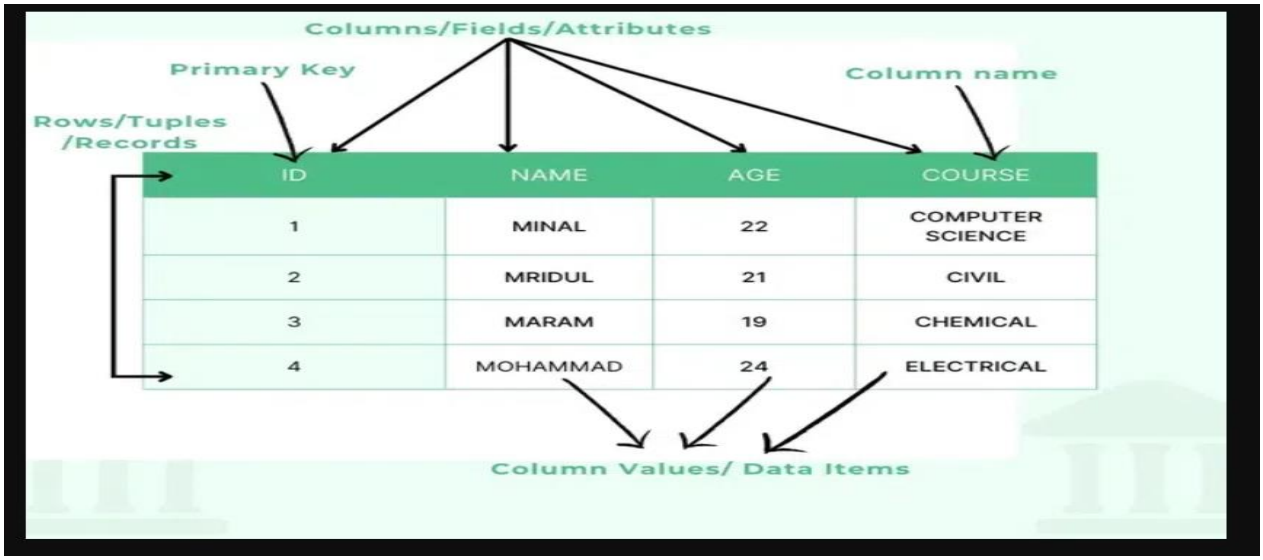
Ex:

```
<?xml version="1.0" encoding="UTF-8"?>
- <EmployeeData>
  - <employee id="34594">
      <firstName>Heather</firstName>
      <lastName>Banks</lastName>
      <hireDate>1/19/1998</hireDate>
      <deptCode>BB001</deptCode>
      <salary>72000</salary>
    </employee>
  - <employee id="34593">
      <firstName>Tina</firstName>
      <lastName>Young</lastName>
      <hireDate>4/1/2010</hireDate>
      <deptCode>BB001</deptCode>
      <salary>65000</salary>
    </employee>
</EmployeeData>
```

Retlational Database

RDBMS stands for Relational Database Management Systems. It is a program that allows us to create, delete, and update a relational database. A Relational Database is a database system that stores and retrieves data in a tabular format organized in the form of rows and columns.

vidyasagar.bandaluppi@gmail.com

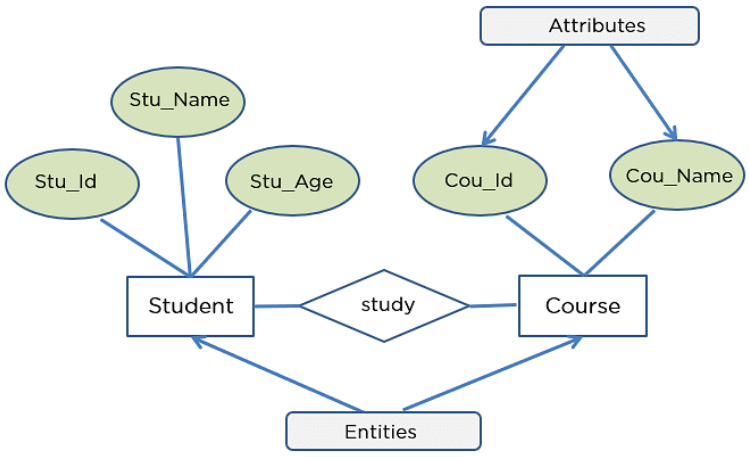


vidyasagar.bandaluppi@gmail.com

Entity Relationship Diagrams

- An Entity Relationship Diagram is a diagram that represents relationships among entities in a database. It is commonly known as an ER Diagram.
- An ER Diagram in DBMS plays a crucial role in designing the database.
- An Entity Relationship Diagram (ER Diagram) pictorially explains the relationship between entities to be stored in a database.
- Fundamentally, the ER Diagram is a structural design of the database.
- It acts as a framework created with specialized symbols for the purpose of defining the relationship between the database entities.
- ER diagram is created based on three principal components: entities, attributes, and relationships.

Eg.,



What is an ER Diagram?

An Entity Relationship Diagram (ER Diagram) pictorially explains the relationship between entities to be stored in a database. Fundamentally, the ER Diagram is a structural design of the database. It acts as a framework created with specialized symbols for the purpose of defining the relationship between the database entities. ER diagram is created based on three principal components: entities, attributes, and relationships.

The following diagram showcases two entities - Student and Course, and their relationship. The relationship described between student and course is many-to-many, as a course can be opted by several students, and a student can opt for more than one course. Student entity possesses attributes - Stu_Id, Stu_Name & Stu_Age. The course entity has attributes such as Cou_ID & Cou_Name.

What is an ER Diagram?

An Entity Relationship Diagram (ER Diagram) pictorially explains the relationship between entities to be stored in a database. Fundamentally, the ER Diagram is a structural design of the database. It acts as a framework created with specialized symbols for the purpose of defining the relationship between the database entities. ER diagram is created based on three principal components: entities, attributes, and relationships.

The following diagram showcases two entities - Student and Course, and their relationship. The relationship described between student and course is many-to-many, as a course can be opted by several students, and a student can opt for more than one course. Student entity possesses attributes - Stu_Id, Stu_Name & Stu_Age. The course entity has attributes such as Cou_ID & Cou_Name.

What is an ER Model?

vidyasagar.bandaluppi@gmail.com

An Entity-Relationship Model represents the structure of the database with the help of a diagram. ER Modelling is a systematic process to design a database as it would require you to analyze all data requirements before implementing your database.

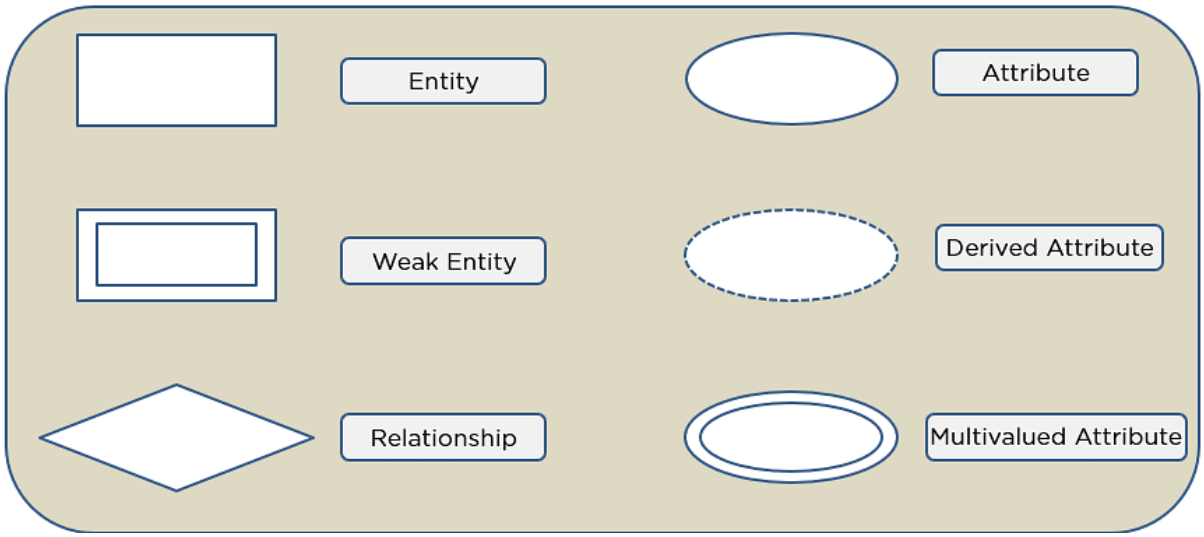
Why Use ER Diagrams in DBMS?

- ER Diagram helps you conceptualize the database and lets you know which fields need to be embedded for a particular entity
- ER Diagram gives a better understanding of the information to be stored in a database
- It reduces complexity and allows database designers to build databases quickly
- It helps to describe elements using Entity-Relationship models
- It allows users to get a preview of the logical structure of the database

Symbols Used in ER Diagrams

- Rectangles: This Entity Relationship Diagram symbol represents entity types
- Ellipses: This symbol represents attributes
- Diamonds: This symbol represents relationship types
- Lines: It links attributes to entity types and entity types with other relationship types
- Primary key: Here, it underlines the attributes
- Double Ellipses: Represents multi-valued attributes

vidyasagar.bandaluppi@gmail.com



Components of ER Diagram

You base an ER Diagram on three basic concepts:

- Entities
    - Weak Entity
- Attributes
    - Key Attribute
    - Composite Attribute
    - Multivalued Attribute
    - Derived Attribute
- Relationships
    - One-to-One Relationships
    - One-to-Many Relationships
    - Many-to-One Relationships
    - Many-to-Many Relationships

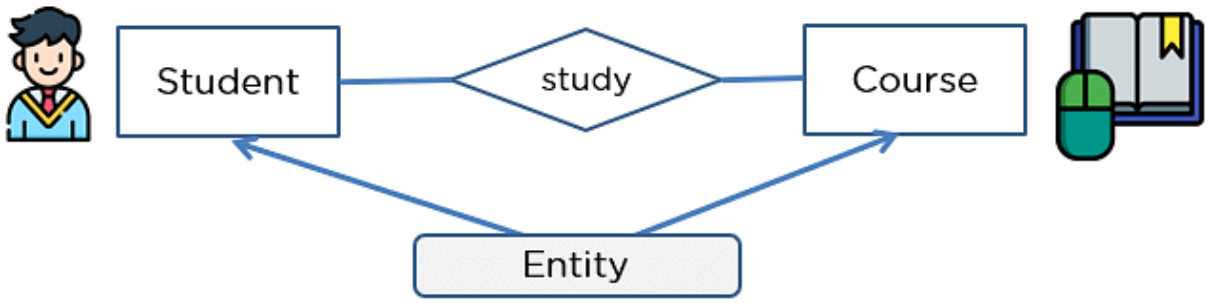
vidyasagar.bandaluppi@gmail.com

Entities

An entity can be either a living or non-living component.

It showcases an entity as a rectangle in an ER diagram.

For example, in a student study course, both the student and the course are entities.

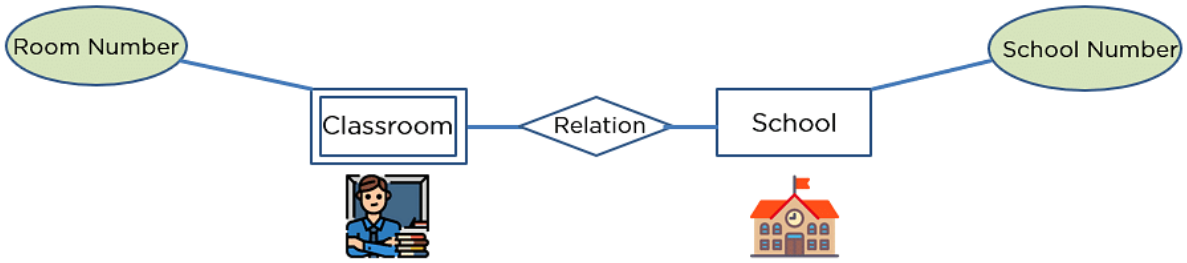


Weak Entity

An entity that makes reliance over another entity is called a weak entity

You showcase the weak entity as a double rectangle in ER Diagram.

In the example below, school is a strong entity because it has a primary key attribute - school number. Unlike school, the classroom is a weak entity because it does not have any primary key and the room number here acts only as a discriminator.

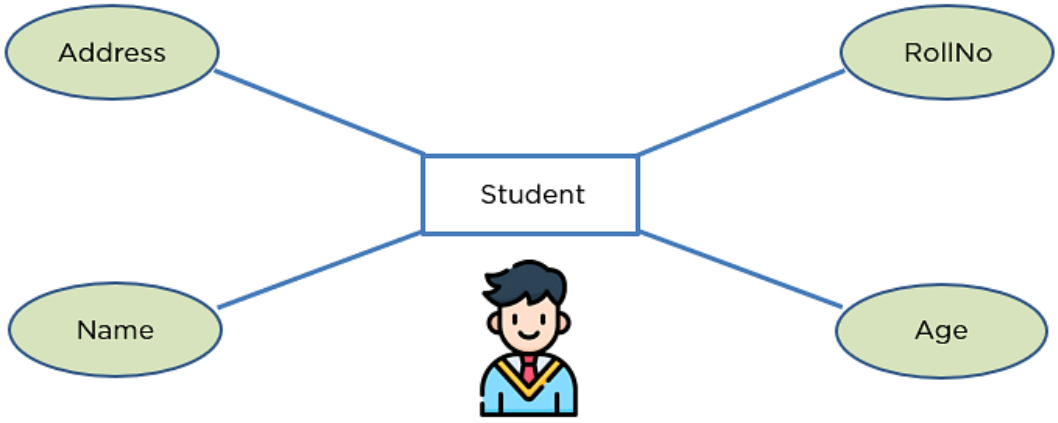


vidyasagar.bandaluppi@gmail.com

Attribute

An attribute exhibits the properties of an entity.

You can illustrate an attribute with an oval shape in an ER diagram.

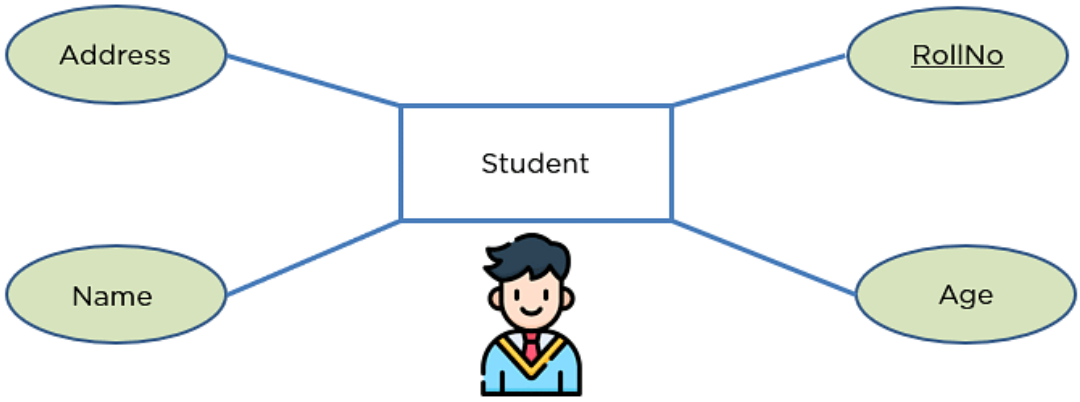


Key Attribute

Key attribute uniquely identifies an entity from an entity set.

It underlines the text of a key attribute.

For example: For a student entity, the roll number can uniquely identify a student from a set of students.

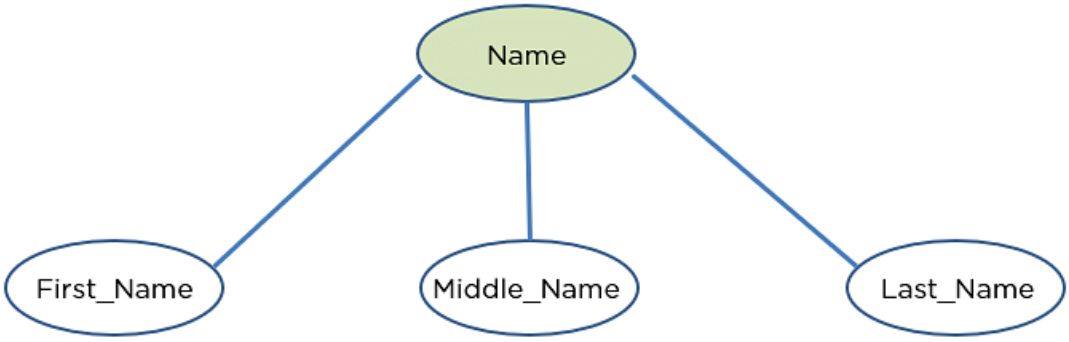


vidyasagar.bandaluppi@gmail.com

Composite Attribute

An attribute that is composed of several other attributes is known as a composite attribute.

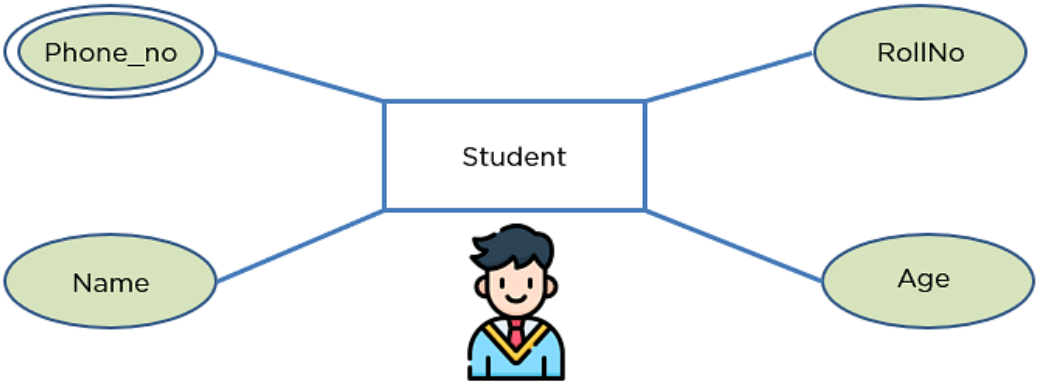
An oval showcases the composite attribute, and the composite attribute oval is further connected with other ovals.



Multivalued Attribute

Some attributes can possess over one value, those attributes are called multivalued attributes.

The double oval shape is used to represent a multivalued attribute.

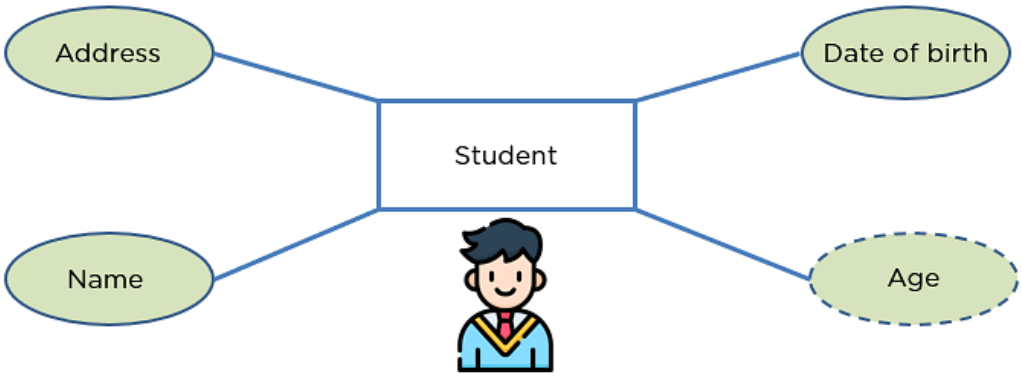


Derived Attribute

An attribute that can be derived from other attributes of the entity is known as a derived attribute.

vidyasagar.bandaluppi@gmail.com

In the ER diagram, the dashed oval represents the derived attribute.

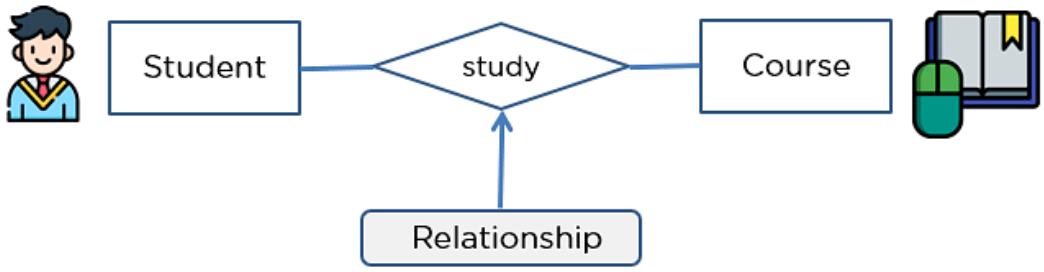


Relationship

The diamond shape showcases a relationship in the ER diagram.

It depicts the relationship between two entities.

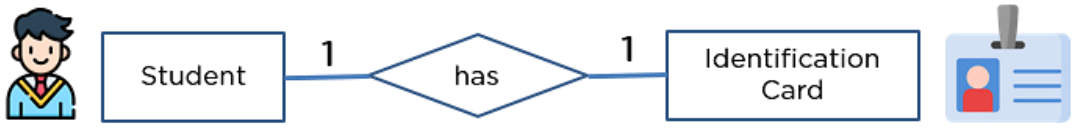
In the example below, both the student and the course are entities, and study is the relationship between them.



One-to-One Relationship

When a single element of an entity is associated with a single element of another entity, it is called a one-to-one relationship.

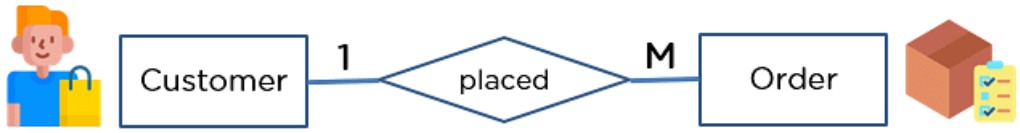
For example, a student has only one identification card and an identification card is given to one person.



One-to-Many Relationship

When a single element of an entity is associated with more than one element of another entity, it is called a one-to-many relationship

For example, a customer can place many orders, but an order cannot be placed by many customers.

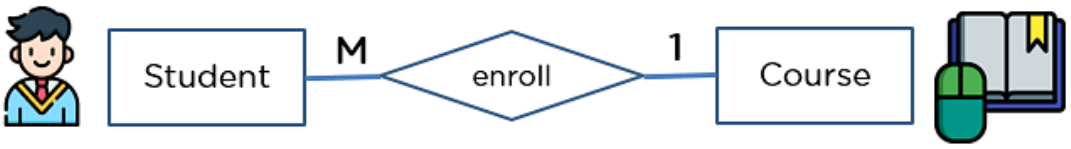


Many-to-One Relationship

When more than one element of an entity is related to a single element of another entity, then it is called a many-to-one relationship.

For example, students have to opt for a single course, but a course can have many students.

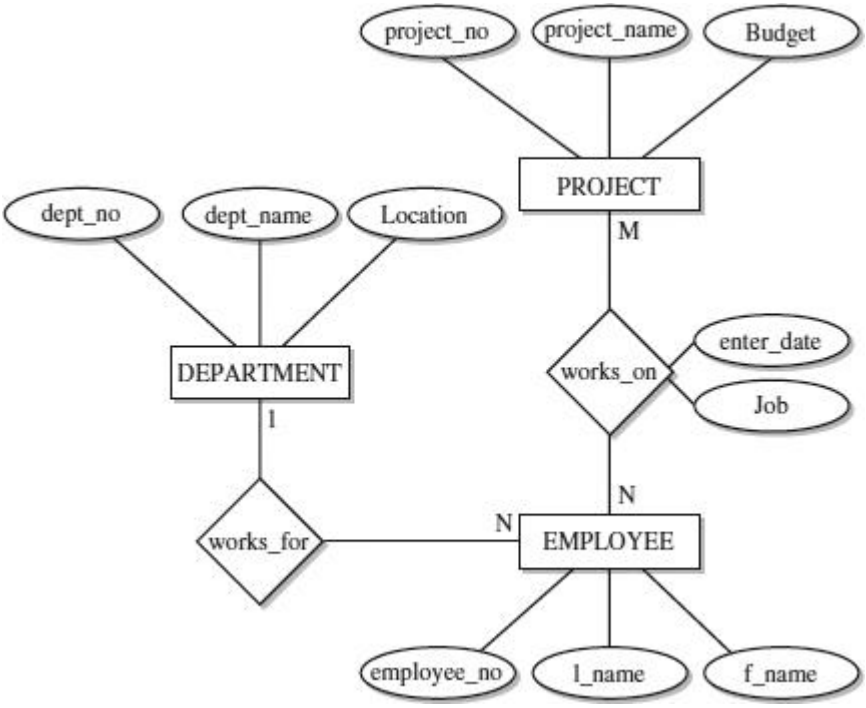
vidyasagar.bandaluppi@gmail.com



Many-to-Many Relationship

When more than one element of an entity is associated with more than one element of another entity, this is called a many-to-many relationship.

For example, you can assign an employee to many projects and a project can have many employees.



vidyasagar.bandaluppi@gmail.com